A Two-Step Chunk-Based Algorithm for Offloading Streaming Traffic through a Vehicular Cloud

Luigi Vigneri
EURECOM
Biot, France
luigi.vigneri@eurecom.fr

Thrasyvoulos Spyropoulos
EURECOM
Biot, France
thrasyvoulos.spyropoulos@eurecom.fr

Chadi Barakat
INRIA Sophia Antipolis
Valbonne, France
chadi.barakat@inria.fr

***Abstract*—Using vehicles equipped with small caches as small base stations has recently been proposed as an interesting middle ground between caching at fixed base stations (which has higher CAPEX/OPEX), and caching at user devices (which has resource limitations). A typical problem in this setup is which content to store in which vehicles. The correct answer depends on the application. Indeed, if the stored content will be streamed (not downloaded), then this offers a natural *delay tolerance*: latter parts of the content do not need to be downloaded immediately from expensive links (e.g., macro-cells), but could be fetched from encountered vehicles cheaply. In an earlier work, we formulated a related optimal cache allocation problem, in which the proposed solution stores a content in its entirety. In light of recent statistics suggesting that different parts of a content (e.g., YouTube clips) are not watched equally frequently, this method is suboptimal. In this paper, we thus consider per-chunk allocation, and propose a simple two-step heuristic that first allocates the vehicular cloud capacity among content items, then efficiently distributes the capacity for a specific content among its chunks. Trace-driven simulation results suggest that chunk-based allocation can lead to considerable gains.**

I. Introduction

Driven by the diffusion of multimedia applications (e.g., Netflix, YouTube), mobile data traffic has seen an exponential growth in the last few years [1]. To keep up with this demand, increasing the number of small base stations (densification) and caching popular content locally, at these base stations or even user devices, are seen as necessary steps [2]. In our recent work, we have proposed the use of private or public vehicles (e.g., cars, buses, taxis) acting as both *mobile* small cells (SCs) and storage points that can be used to offload video streaming traffic [8]. Vehicles bring fundamental advantages compared to static SCs: (i) they are widespread, even in locations where cellular infrastructure is not that advanced; (ii) they can be equipped with storage, communication, and computational capabilities at a lower cost than SCs; (iii) they have few resource limitations, compared to user devices. We refer to such a storage cloud as *vehicular cloud*.

Although caching has recently been considered as a means of improving spectral efficiency and managing interference (see, e.g., the seminal work of [6]), the typical caching problem in the above setup is which content to store in which (vehicular) cache. In this paper, we assume "content streaming" as the application and consider a heterogeneous network setup: a user can download video content chunks into its playout buffer either from nearby vehicles, if these have the needed content stored, or through the traditional cellular network, e.g., from a macro-cell.

The use of such inexpensive "helpers" for downloading content has already been considered in the past [3]. Furthermore, researchers have suggested introducing delay tolerance to offload more traffic when downloading content [8], [10]. However, note that *video streaming offers some intrinsic delay tolerance "for free"*: a user can immediately start the video playout fetching chunks from the infrastructure (without waiting for potential vehicles nearby); once a vehicle carrying chunks of the requested video is in the communication range of the user, (s)he can opportunistically fill the playout buffer, prefetching subsequent chunks while the cache is in range. These bytes are offloaded *without any impact to user experience*. The objective is then to optimally allocate the available vehicular space to minimize the amount of time the playout buffer will be empty (thus necessitating the use of the main cellular infrastructure).

The above problem has considerable complexity as the optimal policy depends on mobility statistics between vehicles and users, content popularities and sizes, etc. In earlier work, we have modeled the playout buffer dynamics as a series of two queues, and proposed a content allocation policy [8]. That policy, based on the above parameters, decides how many vehicles should store a given content. However, a potential caveat in this approach is that content is stored in its entirety in every such car. In light of the viewing behavior of users in many popular video platforms [5], this policy becomes suboptimal: if most users tend to watch, for example, the first part of a video, but many of them seem to skip the second half, then storing both halves in the same number of vehicles wastes space, and the popular parts should be replicated in more caches. At the same time, early chunks have a smaller chance to be downloaded from a vehicle than latter chunks, due to the fact that the latter will be needed later on. This creates an interesting tradeoff between two, possibly contradicting "forces", when trying to decide the optimal number of cached copies in this context not per content, but per chunk.

The rest of the paper is structured as follows:

- First, we introduce the system model and we formulate an optimization problem to maximize the number of chunks offloaded through the vehicular cloud (Sec. II).

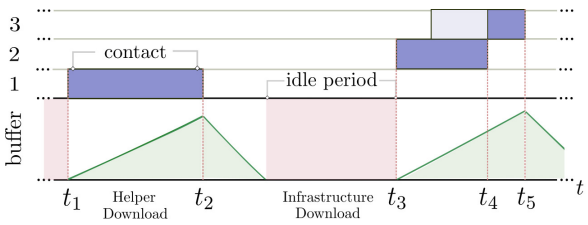


Fig. 1: Sequence of contacts with three caches (above), and amount of data in end user buffer over time (below, in green).

- Then, we review our queueing-based approach to model the playout buffer dynamics, and show how this can be used to infer an efficient *per-content* allocation policy (Sec. III-A).
- Next, based on this result, we propose a two-phase per-chunk allocation policy, where the available space is allocated first among different content items (according to the previous policy), then the space for a specific content is allocated among its chunks, taking into account popularity differences (Sec. III-B).
- Finally, we validate our theoretical results using real traces for content popularity and vehicle mobility (Sec. IV) and we conclude our paper (Sec. V).

II. Model and problem formulation

A. Network model

Nodes: We consider a network with three types of nodes:

- *Infrastructure* ($\mathcal{I}$). Base stations. They provide full coverage, and can serve any content request.
- *Helpers* ($\mathcal{H}$). Vehicles such as cars, buses, trucks, etc., where $|\mathcal{H}| = h$. They store popular content and serve user requests at low cost.
- *Users* ($\mathcal{U}$). Mobile devices such as smartphones, tablets or netbooks. They request (non-live) video content for streaming to $\mathcal{H}$ and $\mathcal{I}$ nodes.

Communication protocol: When a $\mathcal{U}$ node is in range of (at least) an $\mathcal{H}$ node that stores the requested content, the next immediate chunks not yet in the playout buffer are downloaded at low cost at mean rate r_H. This mean rate can be easily inferred from mobility statistics and download rate distribution (*helper download*). On the other hand, when a $\mathcal{U}$ node is not in range of an $\mathcal{H}$ node that stores the requested content *and* its playout buffer is (almost) empty, new chunks are downloaded from the infrastructure at a mean rate r_I until another $\mathcal{H}$ node storing the content is encountered. However, if the playout buffer is *not* empty, no chunks are downloaded from $\mathcal{I}$ until the buffer empties (*infrastructure download*). Chunks in the playout buffer are consumed at a mean viewing *playout* rate r_P. An example of the communication protocol operation is in Fig. 1.

Additional assumptions:

A.1 - Content. Let $\mathcal{K}$ be the set of all possible content items that users might request, with $|\mathcal{K}| = k$. Each content $i \in \mathcal{K}$ consists of a number of small chunks s, and has a popularity value ϕ_i measured as the number of requests within a seeding time window from all users and all cells. Further, each chunk has a popularity $\phi_i \cdot \theta_j$ where $\theta_j \in (0, 1]$ is the normalized internal popularity of chunk j. The number of vehicles that carry chunk j of content i is indicated by x_{ij}. Furthermore, let c be the maximum number of chunks that each vehicle can carry, such that $c \ll k \cdot s$.

A.2 - Mobility model. We assume that the inter-meeting times between a user and a vehicle are IID random variables characterized by a known *generic* distribution with mean rate λ. Contact durations are drawn from another *generic* distribution with mean $\mathbf{E}[D]$.

A.3 - Download rates. We assume $r_I = r_P + \epsilon$ ($\epsilon > 0$ small) to limit the access to the cellular infrastructure to the minimum required to ensure smooth playout (for simplicity, we assume $\epsilon = 0$). We further assume r_I and r_H to be larger than r_P to guarantee uninterrupted streaming. This is a reasonable assumption due to the reduced communication distance: scenarios where r_I (and/or r_H) are lower than the playout rate require initial buffering which is known to significantly degrade QoE [4].

The notation used in the paper is summarized in Table I.

TABLE I: Main notation used in the paper.

x_{ij}	Number of replicas of chunk j for content i
a_i	Number of replicas for content i
k	Number of content in the catalogue
ϕ_i	Number of requests for content i
θ_j	Normalized internal popularity of chunk j
s	Number of chunks per content
c	Maximum number of chunks per vehicle
λ	Mean inter-meeting rate between $\mathcal{U}$ and $\mathcal{H}$ nodes
$\mathbf{E}[D]$	Mean contact duration
h	Number of vehicles
r_P	Mean viewing playout rate
r_H	Mean download rate from $\mathcal{H}$ nodes

B. Problem formulation

Given the above assumptions, the goal of an operator is to minimize the amount of bytes downloaded from the cellular infrastructure for all chunks by appropriately choosing the control vector $\mathbf{x} = \{x_{ij}\}$ that denotes the number of replicas of each chunk across the vehicular cloud. This is captured in the following:

Problem 1. *Consider the system model above. The solution to the following problem maximizes the number of chunks offloaded through the vehicular cloud:*

$$\underset{\mathbf{x} \in X}{\textit{maximize}} \quad \sum_{i=1}^{k} \sum_{j=1}^{s} \phi_i \cdot \theta_j \cdot \Psi_j(x_{ij}), \tag{1}$$

$$\textit{subject to} \quad \sum_{i=1}^{k} \sum_{j=1}^{s} x_{ij} \leq c \cdot h, \tag{2}$$

where $X \triangleq \{\mathbf{a} \in \mathbb{N}^{k \times s} \mid 0 \leq a \leq h\}$ *is the feasible region for the control variable* $\mathbf{x}$*, and* Ψ_j *is the probability that chunk*

j is successfully downloaded from the vehicular cloud, which only depends on how late a chunk is in the video, and on its allocation (i.e., the number of replicas of the chunk).

The objective function counts the number of chunks successfully offloaded through the vehicular cloud in a seeding time window. For each chunk, this is equivalent to the chunk popularity $\theta_j \cdot \phi_i$ times the probability Ψ_j. Eq. (2) makes explicit the capacity constraint[1].

III. Per-chunk allocation

The number of chunks offloaded from the vehicular cloud depends on the number of replicas $\mathbf{x}$, on the vehicles mobility, on the cache size, on the content download rate, and on the number of vehicles. Trying to deal with all these parameters at once makes the problem non-tractable. Rather, we follow a two-step approach: first, we compute a mean replication factor a_i *per-content* in Sec. III-A; then, we modify the internal allocation for a content per chunk (i.e., chunk popularity and intrinsic delay tolerance of different chunks) in Sec. III-B.

A. Optimal per-content allocation

We first compute the optimal allocation *per-content* when the three following assumptions are met:

(i) fractional storage is not allowed, i.e., $x_{ij} = a_i \ \forall i \in \mathcal{K}$;
(ii) all chunks of a given content have the same popularity, i.e., $\theta_{ij} = 1, \ \forall i, j$;
(iii) chunks of a content are downloaded in the same order with which they are played out.

According to these additional assumptions, we formulate a per-content allocation problem where the goal of the operator is to minimize the amount of bytes downloaded from the cellular infrastructure $\mathbf{E}[W_i]$, by appropriately choosing the number of replicas per content a_i:

Problem 2. *The solution of the following problem minimizes the expected number of bytes downloaded from the cellular infrastructure when fractional storage is not allowed:*

$$\underset{a_i \in \{0,1,\ldots,h\}}{\textit{minimize}} \ \sum_{i=1}^{k} \phi_i \cdot \mathbf{E}[W_i], \tag{3}$$

$$\textit{subject to} \quad \sum_{i=1}^{k} a_i \leq \frac{c}{s} \cdot h. \tag{4}$$

A natural modeling of this scenario can be performed through queueing theory. A contact between a device and a vehicle corresponds to a new arrival in the playout buffer of the device. A new arrival brings a random amount of new bytes that depends on the contact duration with that vehicle. In this work, we model the playout buffer as a bulk G^Y/D/1 queue[2] where new bytes arrive in *bulks* when a vehicle storing the requested content is encountered, and are consumed at a mean playout rate r_P (small square of Fig. 2). In *busy* urban environments, if a user is downloading video i from vehicle A and the connection is lost, the user could just keep downloading from another node B storing i, also in range. Hence, as long as there is *at least* one cache with a copy within range, the user will keep downloading content i at mean rate r_H. We can model these overlapping contacts with an extra G/G/∞ queue in front of the playout queue (as shown in Fig. 2). New vehicles arrive in the G/G/∞ queue with rate $\lambda \cdot x_i$, each staying for a random service time (corresponding to a contact duration with mean $\mathbf{E}[D]$) and independently of other cars. The number of jobs in the G/G/∞ queue is the number of vehicles concurrently within range of the user. Hence, it is easy to see that: (i) the beginnings of busy periods of the G/G/∞ queue correspond to new bulk arrivals in the playout buffer, and (ii) the mean duration of such busy periods, multiplied by r_H, corresponds to the mean bulk size per arrival.

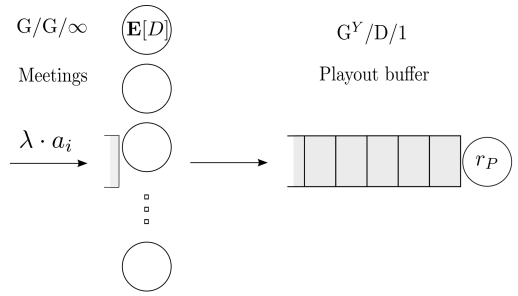


Fig. 2: Proposed queueing model for the playout buffer.

Lemma 3.1. *Assume s large. Assume further that a_i replicas are cached for content i. Then, the expected number of bytes downloaded from the cellular infrastructure $\mathbf{E}[W_i]$ is equal to*

$$\mathbf{E}[W_i] = s \cdot \left[1 - \left(1 - e^{-\lambda \cdot \mathbf{E}[D] \cdot a_i}\right) \cdot \frac{r_H}{r_P}\right]. \tag{5}$$

Proof. The proof can be found in [9]. □

When we replace Eq. (5) in the formulation of Problem (2), it is straightforward to see that we get an NP-hard combinatorial problem. In [8] we have solved a continuous relaxation using the Karush-Kuhn-Tucker (KKT) conditions. In this specific scenario, it can be easily proven that the optimal number of replicas is proportional to the logarithm of the content popularity.

B. Per-chunk adaptation

In the previous subsection, we have made the simplifying assumption that either all chunks or no chunks of a content must be cached in a vehicle. Said otherwise, every chunk of a content must have the same number of replicas. Yet, as mentioned earlier, two opposing "forces" call for a per-chunk optimization policy: (i) it is perhaps wasteful to cache too many of the early chunks as there might not be enough time to fetch these chunks anyway; (ii) if early chunks have higher

[1] We do not need to put extra constraints for individual car capacities, because we assume that all the vehicles are statistically identical and we limit the number of replicas to the number of vehicles h.

[2] G^Y/D/1 describes a queue where arrivals follows a generic distribution with a random variable Y for the number of arrivals at one time, deterministic service time, and a single server.

popularity than latter chunks, the former perhaps deserve more storage space.

To this end, we propose a heuristic algorithm which adapts the optimal per-content solution to an internal allocation which depends on the aforementioned tradeoff. Specifically, we design an allocation policy which takes as input the optimal per-content allocation (of Sec. III-A) and provides a per-chunk allocation. Let us consider content i which is split into s chunks. Assume that the per-content allocation is of a_i copies which corresponds to $s \cdot a_i$ chunks. Let x_{ij} be the number of replicas of chunk j for content i. While in the previous subsection we had $x_{ij} = a_i \ \forall j$, the goal of this section is to find the vector of chunk allocation $\{x_{ij}\}$ that further improves the volume of traffic offloaded. The idea is to reshuffle the copies according to the internal content popularity (i.e., chunk popularity) and to the chunk delay (i.e., when the chunk will start to be played out). We summarize this idea in the following optimization problem:

Problem 3. *Let a_i be the number of replicas of content i. The following optimization problem provides the optimal chunk allocation given the internal popularity θ_j:*

$$\begin{aligned} &\underset{x_{ij}\in\{0,1,\ldots,h\}}{\textit{maximize}} && \sum_{j=1}^{s} \theta_j \cdot \Psi_j(x_{ij}), \\ &\textit{subject to} && \sum_{j} x_{ij} = s \cdot a_i. \end{aligned}$$

The probability Ψ_j depends on the number of replicas, on the mobility statistics and on the time at which a chunk is played out. Although calculating the correct probability is difficult due to these several dependencies, we will provide two approximations of Ψ_j which will be used to compute the per-chunk allocation.

1) Example 1 (proportional): assume Ψ_j depends (linearly) only on the number of replicas. Hence,

$$\Psi_j(x_{ij}) = \frac{1}{h} \cdot x_{ij}.$$

When Ψ_j is defined as above, Problem (3) corresponds to a bounded knapsack problem. In this case, the most popular chunks will have h replicas and the others 0.

2) Example 2 (infinite bandwidth): if a vehicle storing a subset of chunks for content i comes in range to the user, we assume that the user is able to download all the chunks of content i carried by the vehicle. Note that we do not require that this is true in a real setup. We simply say that our policy will assume so, for simplicity (and thus might not always be optimal). This assumption becomes more accurate if, for example, there are many vehicles and many content items, so that each vehicle carries only few chunks per content. If vehicles arrive in the user communication range as a Poisson process, the probability Ψ_j can be rewritten as follows:

$$\Psi_j(x_{ij}) = 1 - \exp\{-\lambda \cdot (j-1) \cdot \tau \cdot x_{ij}\}, \tag{6}$$

where τ is the duration of each chunk in seconds.

Proof. $\Psi_j(x_{ij})$ follows from the probability that any of the vehicles having a copy of chunk j of content i is met within the time tolerance for this chunk which is equal to $(j-1) \cdot \tau$. The contacts between the user and a vehicle are assumed to follow a Poisson process of rate λ. □

Lemma 3.2. *Consider a continuous relaxation of Problem* (3)*. When Ψ_j is described by Eq.* (6)*, the optimal solution is given by*

$$x_{ij}(a_i, j) = \frac{1}{w_j} \cdot \ln\left(\frac{w_j}{\rho(a_i)} \cdot \theta_j\right),$$

where $w_j \triangleq \lambda \cdot (j-1) \cdot \tau$ and $\rho(a_i)$ is an appropriate Lagrange multiplier. Furthermore, due to the box constraints, we assign 0 replicas to low popular chunks or h replicas to high popular chunks.

Proof. Problem (3) is clearly convex since the objective function is a sum of convex functions, the constraints are linear and the domain of the feasible solutions is convex. We solve it by Karush-Kuhn-Tucker (KKT) conditions. For such a problem, this method provides necessary and sufficient conditions for the stationary points to be optimal solutions. □

According to the above discussion, we summarize here the proposed two-step algorithm:

- *Step 1.* Compute a per-content allocation $\{a_i\}$ assuming all chunks have same popularity. The replication factor is the result of the queueing model proposed in Sec. III-A.
- *Step 2.* Step 1 suggests to store $s \cdot a_i$ chunks for content i in the vehicular cloud. Reshuffle these chunks according to chunk popularity and chunk delay as shown, e.g., in Lemma 3.2.

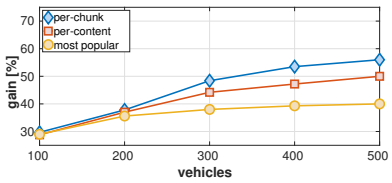
IV. Evaluation

A. Simulation Setup

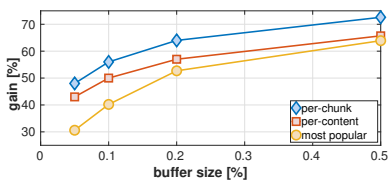
To validate our results, we perform simulations based on real traces of (i) vehicle mobility and (ii) content popularity:

- *Vehicle mobility.* We use the Cabspotting trace [7] to simulate the vehicle behaviour; this trace records the GPS coordinates for 531 taxis in San Francisco for more than 3 weeks with granularity of 1 minute. In order to improve the accuracy of our simulations, we increase the granularity to 10 seconds by linear interpolation.
- *Content.* We infer the number of requests per day from a database with statistics for 100.000 YouTube videos [11]. We suppose each content is split in 10 chunks. To increase the number of simulations and to provide sensitivity analysis for buffer capacity and vehicle density, we limit the number of content to 10.000 selected randomly from the initial set.

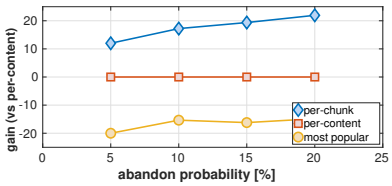
We build a MATLAB simulator as follows: first, we generate a set of requests, and we associate a random location to each one. The number of requests per content per day is given by the YouTube trace. Then, we store chunks in vehicles according to chosen allocation policy. For each request, we simulate the playout of the video; the end user buffer will be opportunistically filled when the vehicular cloud can be contacted,



(a) Offloading gain vs. number of vehicles.



(b) Offloading gain vs. cache size.



(c) Offloading gain vs. abandon rate.

Fig. 3: Performance evaluation.

according to the mobility provided by the Cabspotting trace. The number of chunks downloaded per contact depends on the contact duration and on the distance between user and vehicle. In our simulations, we assume that end users can contact the vehicular cloud within 200 m. The user abandons the playout of the video with some probability: specifically, when a user watches a chunk, she decides to watch the following chunk with probability $1-q$ or to abandon the playout with probability q (we set $q = 0,05$ as default value). We scale down the vehicle storage capacity c to ensure that 0,1% of the catalogue fits in each cache (i.e., 10 entire content items).

We compare the following allocation policies:

- *per-chunk:* allocation policy described in Sec. III-B when Ψ_j is given by Eq. (6);
- *per-content:* allocation policy described in Sec. III-A which considers equal popularity for all chunks of a content;
- *most popular:* store the most popular chunks.

Our main goal in this preliminary evaluation is to highlight the improvements, in terms of number of chunks offloaded, brought by caching per-chunk compared to per-content, in a realistic scenario.

B. Caching Strategy Evaluation

In Fig. 3a we perform sensitivity analysis according to the number of vehicles h in the cloud which varies from 100 to 500. While the number of envisioned connected vehicles in the center of San Francisco is expected to be much larger, the low technology penetration rate analyzed still provides considerable amount of data offloaded. As the number of vehicles increases, the per-chunk policy offloads much more traffic compared to the other policies. For example, with 500 vehicles, this policy can offload almost 60% of the traffic.

Fig. 3b compares different buffer capacities per vehicle. Buffer size is in the range 0,05-0,5% of the catalogue (where $h = 531$). Considerable performance gains (i.e., more than 70% of traffic offloaded) can be achieved with very reasonable storage capacities (i.e., less than 1% of the catalogue). Here the simulations are performed on a set of 10.000 content items, but in a scenario with a larger realistic catalogue, it seems doable to store 0,05-0,5% of the content needed to achieve good savings. E.g., if one considers the entire Netflix catalogue ($\sim$ 3 PB), a buffer capacity of about 3 TB (0,1%) already suffices to offload almost 60% of the traffic.

In Fig. 3c we perform sensitivity analysis according to the probability of viewing the subsequent chunk. We analyse the range 5-20% (i.e., a video is entirely played out from 10 to 60% of times). In this scenario, the per-chunk policy provides a relative improvement between 10 and 25% compared to the per-content policy. Moreover, the relative traffic offloaded by the per-chunk policy improves as the abandonment probability increases. This confirms that reshuffling chunks is an effective strategy to improve traffic offloading. Finally, we believe that a finer per-chunk policy (i.e., introducing a more realistic function for Ψ_j) would provide even larger gains.

V. Conclusion

In this paper, we have focused our study on caching multimedia content such as videos. Video streaming has become dominant in the current Internet traffic [1]. We have formulated a model that accounts for chunk popularity and the fact that later chunks introduce an intrinsic delay tolerance on the content download. Using queueing theory notions, we were able to model the intermittent contacts with the mobile caches to propose a per-chunk allocation algorithm. Finally, we have validated our finding through real trace-based simulations.

References

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update. 2016-2021.

[2] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang. What Will 5G Be? *IEEE Journal on Selected Areas in Communications*, 32(6):1065–1082, June 2014.

[3] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire. Femtocaching: Wireless video content delivery through distributed caching helpers. In *IEEE INFOCOM*, 2012.

[4] T. Hossfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen. Initial delay vs. interruptions: Between the devil and the deep blue sea. In *2012 Fourth International Workshop on Quality of Multimedia Experience*, pages 1–6, July 2012.

[5] S. H. Lim, Y. B. Ko, G. H. Jung, J. Kim, and M. W. Jang. Inter-chunk popularity-based edge-first caching in content-centric networking. *IEEE Communications Letters*, 18(8):1331–1334, Aug 2014.

[6] M. A. Maddah-Ali and U. Niesen. Fundamental limits of caching. *IEEE Transactions on Information Theory*, 60(5):2856–2867, May 2014.

[7] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser. DAD data set epfl/mobility (v. 2009-02-24), 2009.

[8] L. Vigneri, T. Spyropoulos, and C. Barakat. Storage on wheels: Offloading popular contents through a vehicular cloud. In *IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2016.

[9] L. Vigneri, T. Spyropoulos, and C. Barakat. Streaming content from a vehicular cloud. CHANTS '16, pages 39–44. ACM, 2016.

[10] J. Whitbeck, M. Amorim, Y. Lopez, J. Leguay, and V. Conan. Relieving the wireless infrastructure: When opportunistic networks meet guaranteed delays. In *IEEE WoWMoM*, pages 1–10, June 2011.

[11] M. Zeni, D. Miorandi, and F. De Pellegrini. YOUStatAnalyzer: a tool for analysing the dynamics of YouTube content popularity. In *Proc. 7th International Conference on Performance Evaluation Methodologies and Tools (Valuetools, Torino, Italy, December 2013)*, Torino, Italy, 2013.